19/10/2006

ALPAGE

Peers, how many are we?

System size estimation in large scale overlays

Ayaldi Ganesh – Microsoft Research

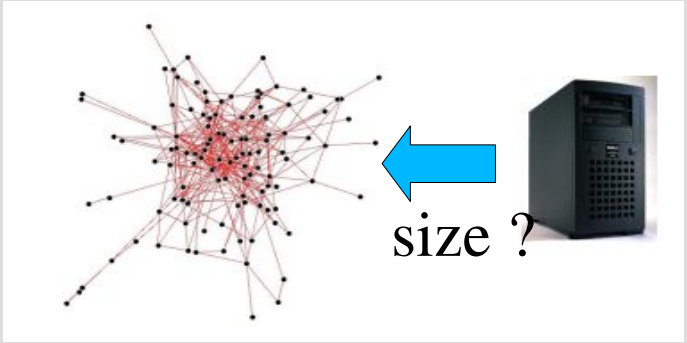
Anne-Marie Kermarrec - INRIA/IRISA

Erwan Le Merrer – FTR&D/IRISA

Laurent Massoulié - Thomson Research

Why count? (Motivations)

- Parameter value setting
- Network monitoring, avoiding hot spot on the requester



- *Focus on generic algorithms*:
  - fully distributed
  - topology independant (connected graph)
  - no need for extra structure / node state

Outline

- Main counting algorithm classes
- Random Tour and Sample&Collide
- Comparisons in static and dymanic networks
- Discussion on improvement
- Summary / Conclusion

Main counting algorithm classes #1: Probabilistic polling class

- init message spread, and probabilistic response to avoid message implosion
- Ex: Hops Sampling (minHopsReporting)

D. Kostoulas, D. Psaltoulis, I. Gupta, K. Birman, A. Demers: **Decentralized**

Schemes for Size Estimation in Large and Dynamic Groups. NCA 2005.

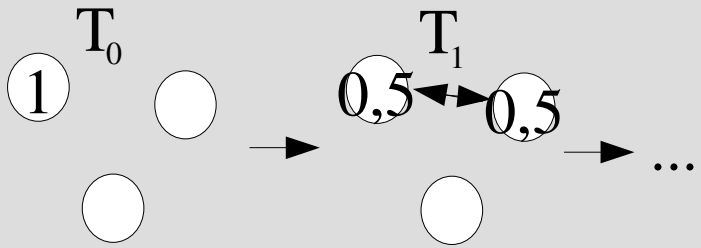
  - gossip based broadcast (to reduce overhead) carrying a distance information from the initiator
  - probabilistic response based on this distance («far» nodes have lower chance to reply back)

Main counting algorithm classes #1: Probabilistic polling class

- In more details:
  - Initiator spreads a message initialised with *hopCount*=0 and its IP address
  - A receiver forwards it to *gossipTo* neighbors (*hopCount*++) for *gossipFor* rounds or until *gossipUntil* messages have been received
  - Once a peer stops gossiping, it sends back a HELLO message to the initiator:
    - with proba. 1 if $hopCount < minHopsReporting$
    - with proba. $\frac{1}{gossipTo^{(hopCount - minHopsReporting)}}$ otherwise
  - The size estimation is computed by the initiator based on the number of peer responses and their distance from the initiator

Main counting algorithm classes #2: Gossip-based aggregation

- M. Jelasity and A. Montresor: **Epidemic-Style Proactive Aggregation in Large Overlay Networks**. ICDCS 2004.
- Idea: «If exactly one node of the system holds a value equal to 1, and all the other values are equal to 0, the average is $\frac{1}{N}$ »



- In more details:
  - The initiator takes the value 1 and starts gossiping (reached peers start with 0)
  - At each time interval each peer chooses a random neighbor and swaps it's local value (push/pull) $newValue = \frac{(localValue + neighbor'sValue)}{2}$
  - At each peer, after an sufficient propagation time $\hat{N} = \frac{1}{value}$

Main counting algorithm classes #3: Random walk based

- Random increasing walk [Bawa et al.]:

Estimating aggregates on a peer-to-peer network. Tech. Rep. 2003.

Here complete graph:

  - start from the smallest node ID
  - pass to its randomly selected neighbor with a greater ID
  - expected path length *l* is $O(\ln N)$ (thus $\hat{N} = e^{l}$)

  But knowledge of the topology needed !

  We now introduce 2 other random walk based techniques

Randour Tour and Sample&Collide

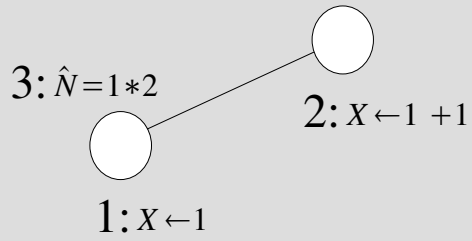
- L. Massoulié, E. Le Merrer, A.-M. Kermarrec, A. Ganesh: **Peer counting and sampling in overlay networks: random walks methods**. PODC 2006.

- Random Tour: new system size estimation with a random walk based algorithm

- Sample&Collide: random walk based peer sampling and birthday paradox reversal

Random Tour

- Algorithm:
  - Initiator *i* initialises a counter value *X* with $\frac{1}{d_i}$
  - until the return to *i*, the counter is forwarded to a neighbor *j* chosen uniformly at random, $X \leftarrow X + \frac{1}{d_j}$
  - At counter return on *i*, $\hat{N} = d_i * X$



- Average overhead $O(N)$

Sample&Collide

- Based on *the birthday paradox*:
  - For $\sqrt{N}$ independant samples from a population of size N, the probability that a pair of samples will have the same value is at least $\frac{1}{2}$
  - Probability 1 is concentrated around $\sqrt{2N}$
- Inverted paradox:
  - sample uniformly at random until a collision is found
  - the system size is computed from the number of used samples

Sample&Collide

- In more details:
  - Unbiased sampling emulates continuous time random walks:
    - T>0 is set by the initiator and sent to a random neighbor
    - The receiver chooses U [0,1] and decrements T by $\frac{-(\log(U))}{d_i}$;
      - if T>0 it then forwards T to a random uniformly selected neighbor
      - otherwise, the current node is the sample (HELLO to the initiator)
  - The number of samples X drawn till the collision is used for the estimate $\hat{N} = \frac{X^2}{2}$
  - The control parameter *l* (accuracy/overhead) refines the estimation based on the number of collisions observed ($\hat{N} = \frac{X^2_l}{2l}$)

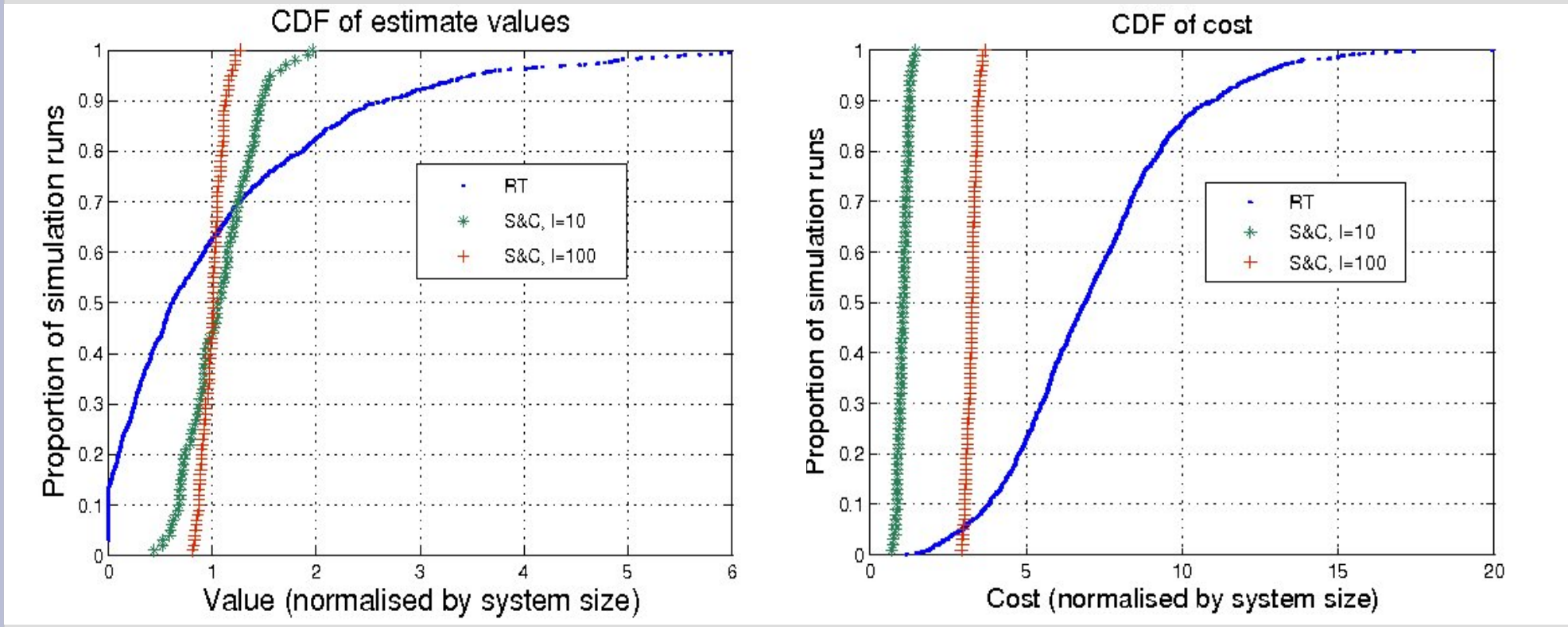
Randour Tour and Sample&Collide

Counting only a fraction of nodes with certain capacities?

(degree>100, bandwith<100Mb/s, functions, ...)

- Random Tour: only increment *x* if the node verifies the condition
- Sample&Collide:
  - Samples are added to the sample list only they owns the property
  - System size estimation, and extrapolation from the percentage of sampled nodes with the property

Randour Tour and Sample&Collide



- RT vs S&C, accuracy and cost

  (100000 nodes, average node degree = 7)

Simulations

- Considered metrics
  - accuracy
  - reactivity
  - overhead
- Algorithms settings
  - HopsSampling: gossipTo=2, gossipFor=gossipUntil=1, minHopsReporting=5
  - S&C: T=10, l=200
- Network settings
  - Simulator based results, no message loss
  - Heterogeneous wiring: 1 to 10 neighbors per peer (7.2 average)

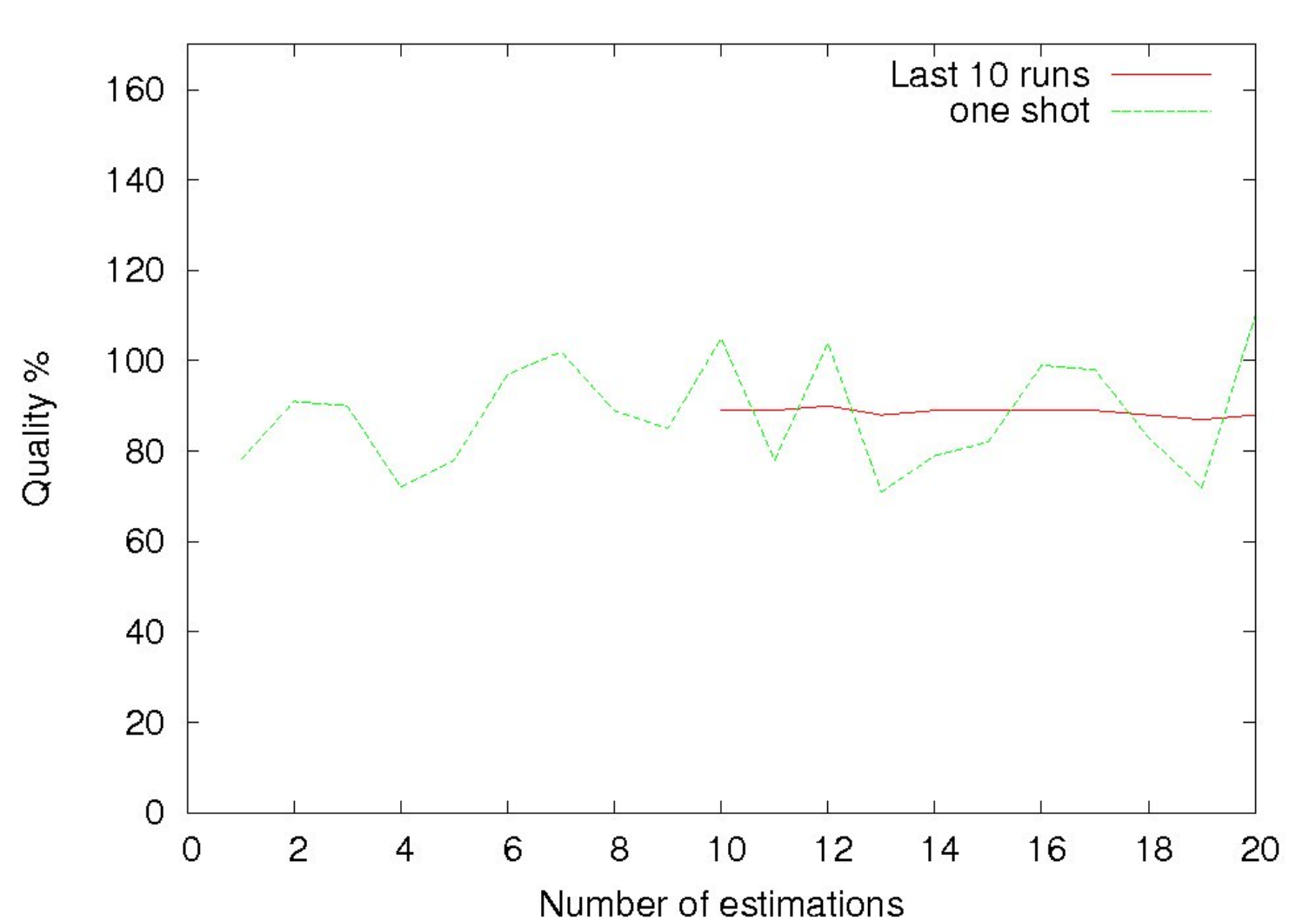
Simulations

Static networks

1,000,000 nodes

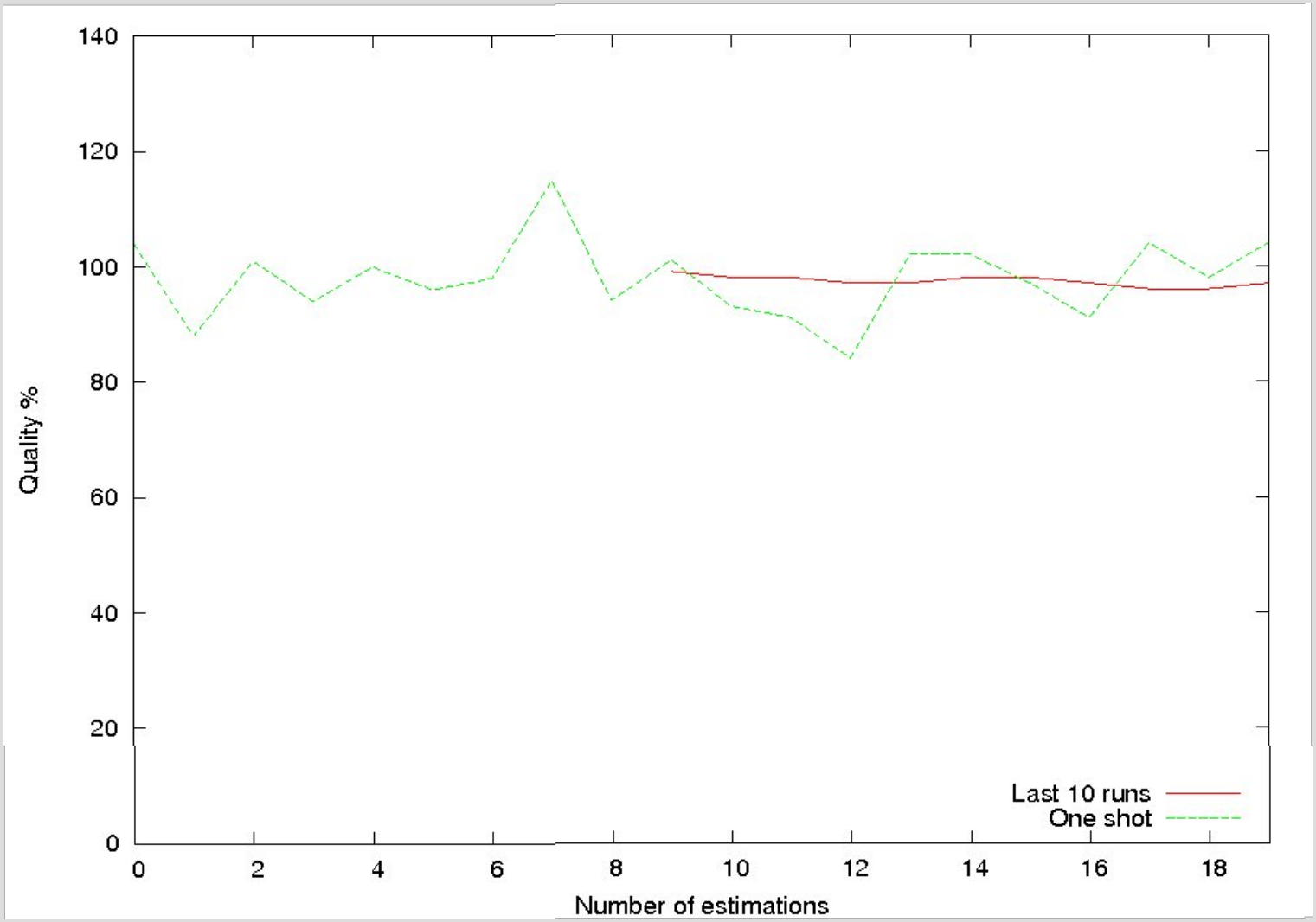
Simulations

Hops Sampling (minHopsReporting)



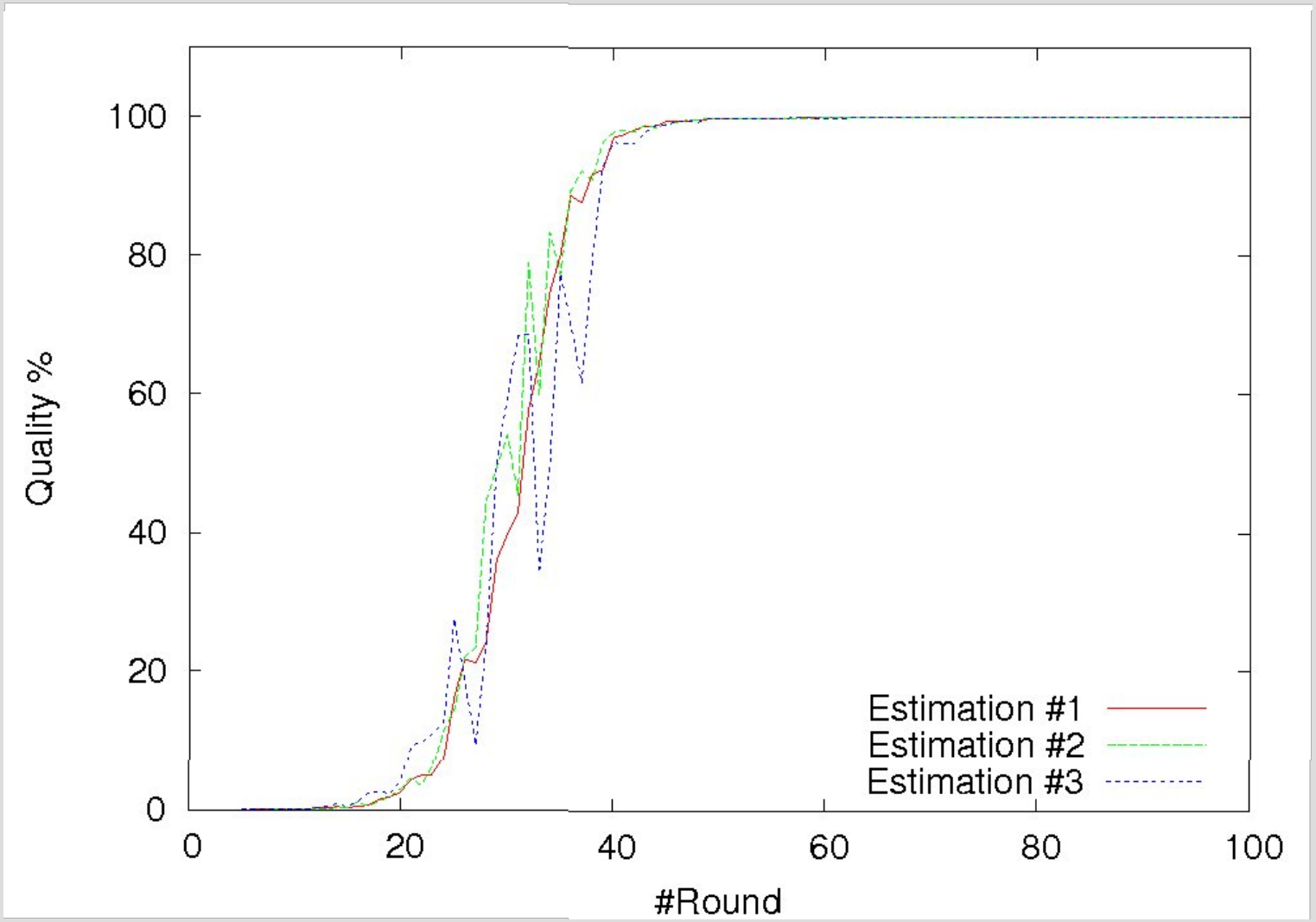
Simulations

Sample&Collide



Simulations

Gossip-based aggregation



Simulations

Dynamic networks

100,000 nodes

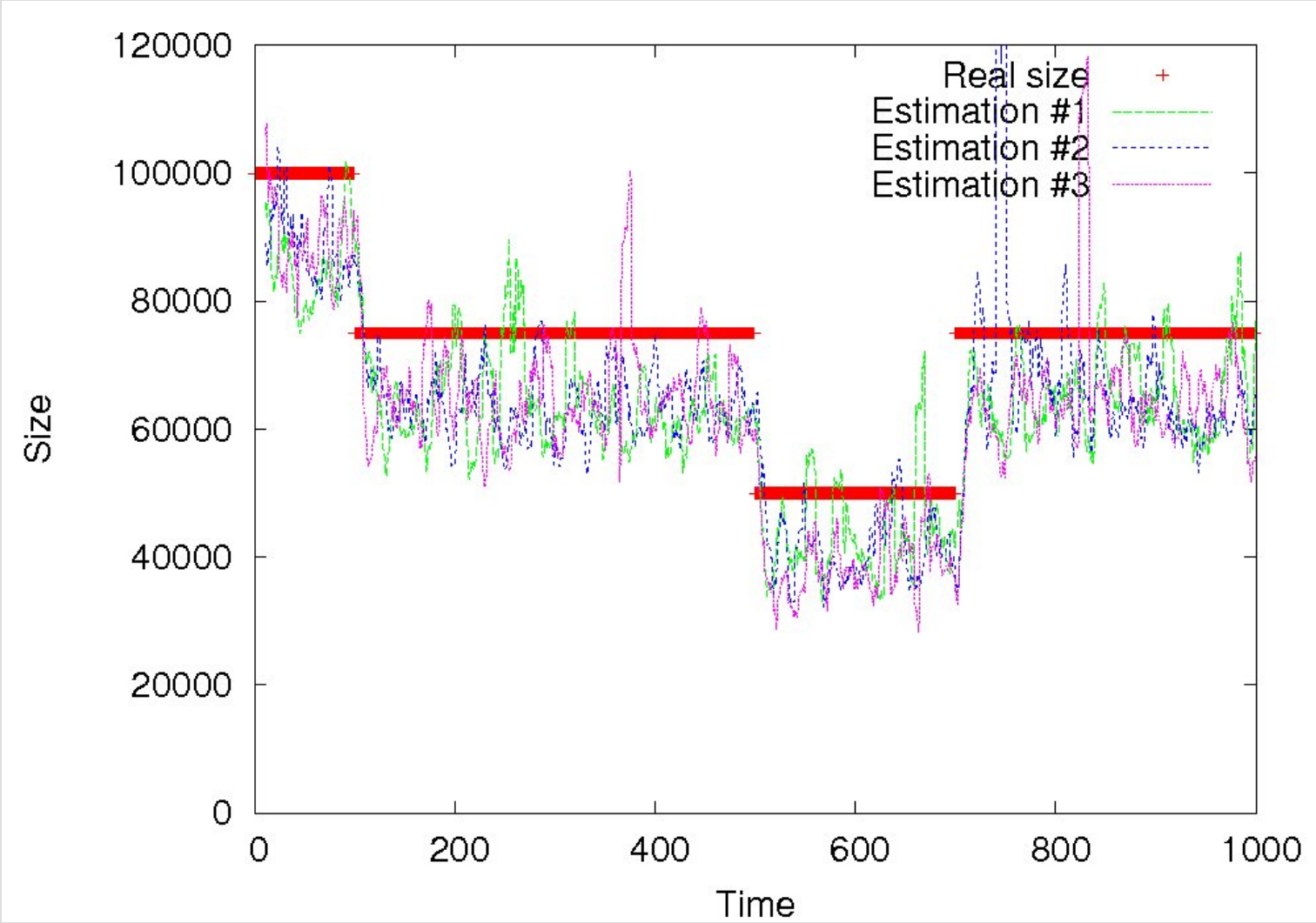
HopsSampling is averaged with the last 10 results

S&C is used non averaged, l=200

Aggregation results after 50 rounds, and continuously restarted

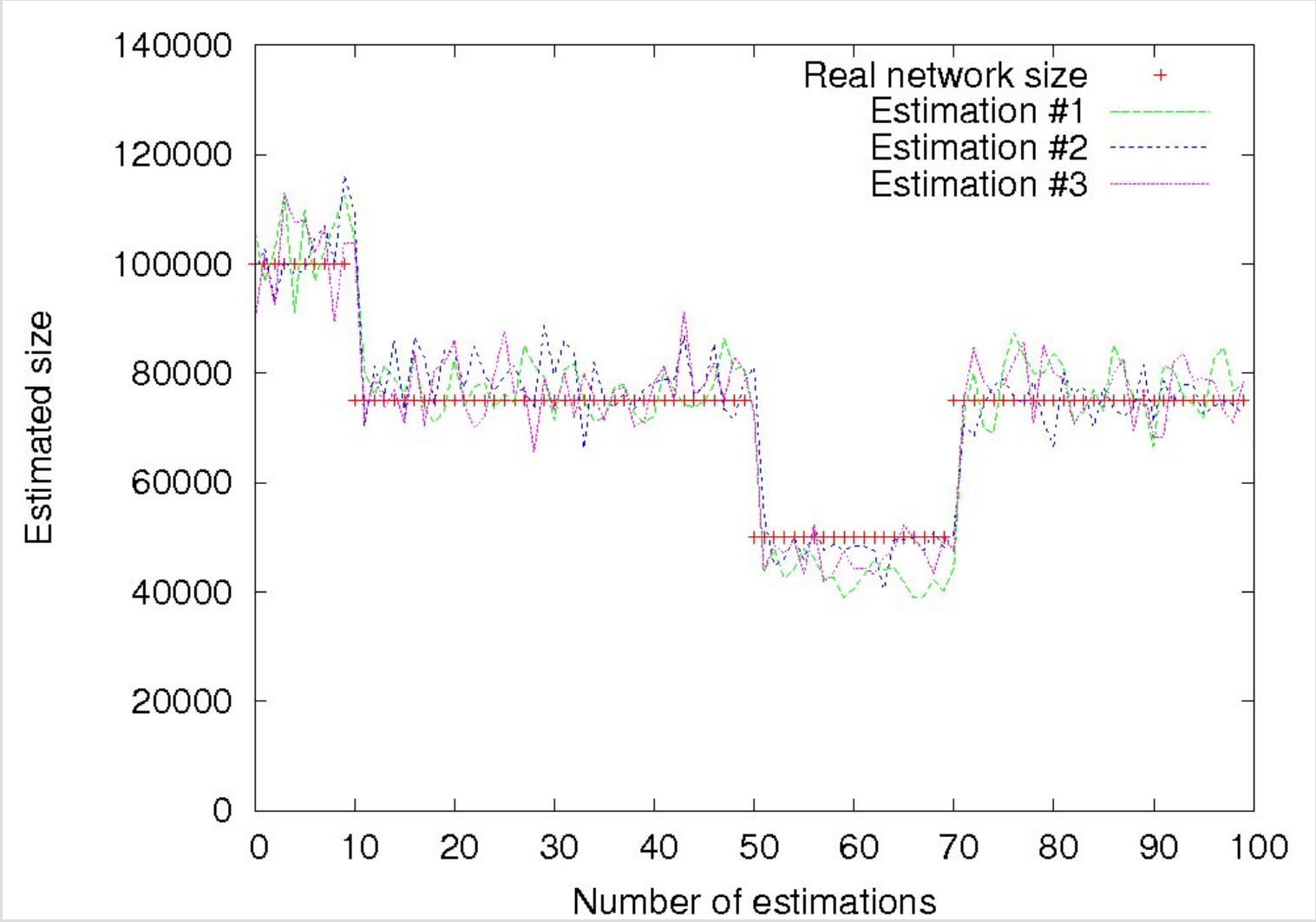
Simulations

Hops Sampling (minHopsReporting)



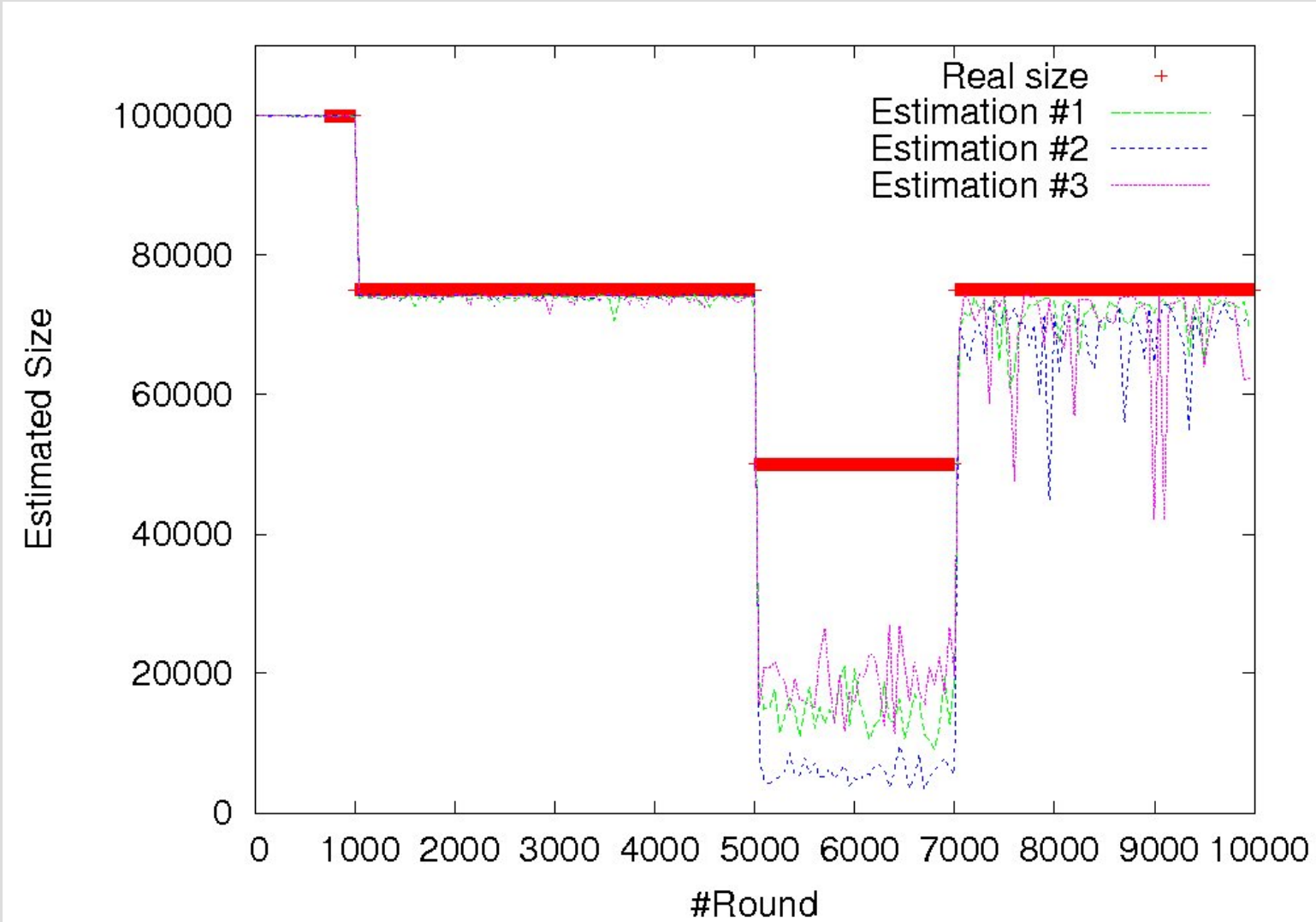
Simulations

Sample&Collide

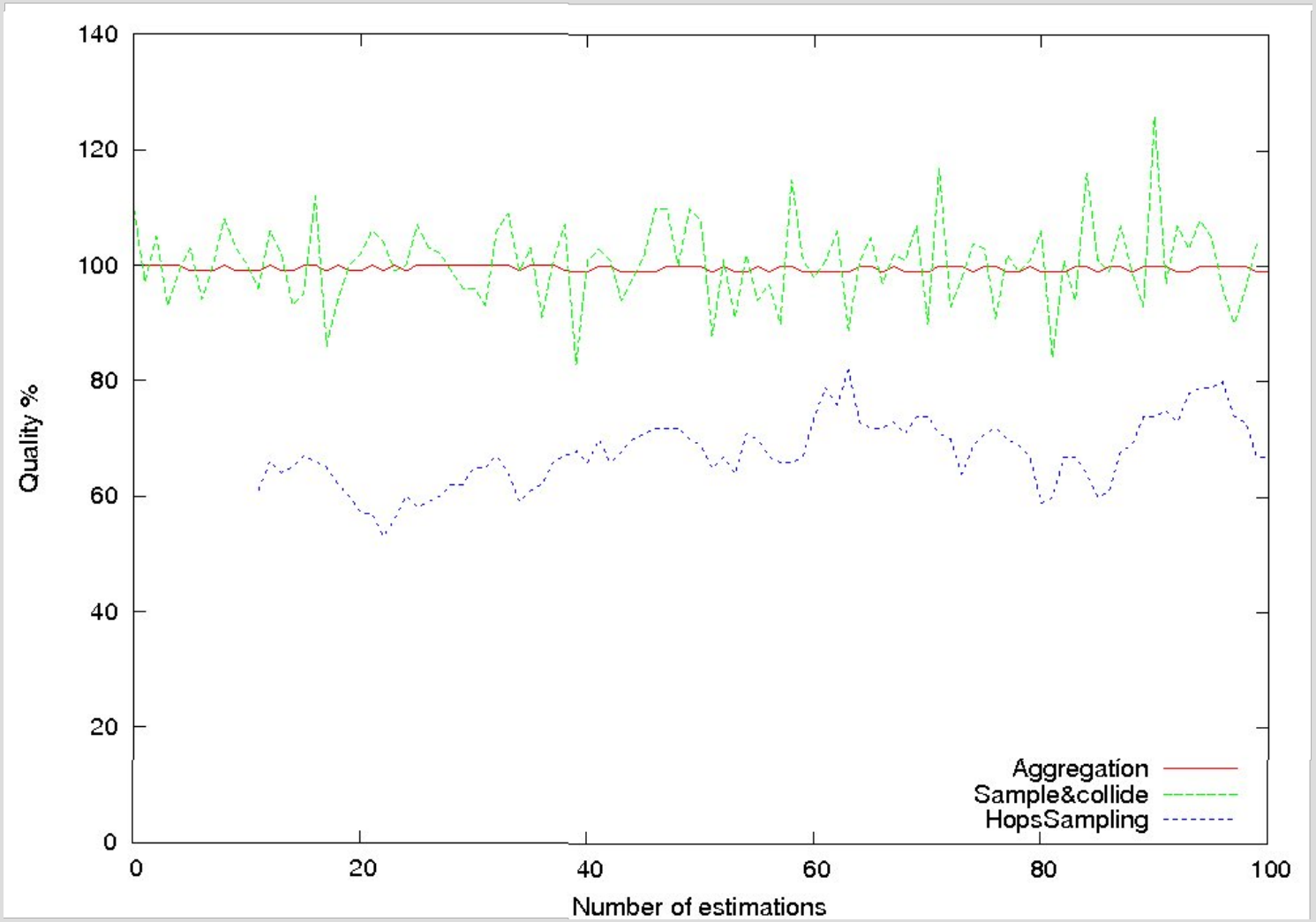


Simulations

Gossip-based aggregation



Simulations



- 100,000 nodes scale free topology $P(k)=k^{-3}$

Simulations

Overhead

Algorithm	*S&C*	*HopsSampling*	*Aggregation*
Parameters	l=200	averaged (last 10 runs)	50 rounds
Accuracy	+/- 10%	- 20%	- 1%
Overhead	0,5 M	2,5 M	10 M
Prediction	$[sampling\ cost] * \sqrt{2lN}$	$Depends\ of\ the\ spreading\ algo + its\ parameters$ $O(N)$	$N * nbRounds * 2$ $O(N \log(N))\ for\ expanders$

- Exemple of accuracy/overhead tradeoff on the 100,000 nodes static network

Discussion

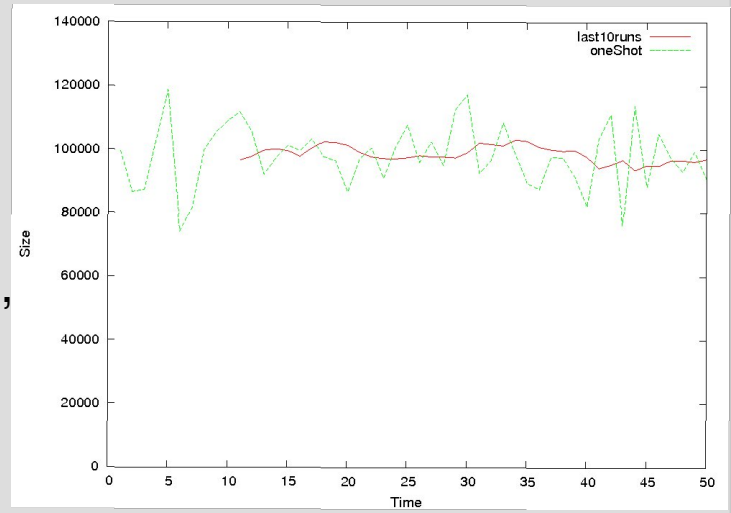
suggested improvements

- HopsSampling inaccuracy:

  Unbiased algorithm: underestimate factor is due to a non optimal gossip message spread

  Well tuned parameters: accurate non averaged results, but needs an idea of the system size!

  Use of more generic broadcast techniques? (higher overhead!)



gossipFor=gossipTo=gossipUntil=5

- Aggregation, number of rounds to wait:

  Instead of using a fix number (how to predict it?), a solution could be to observe the local estimation stabilisation to decide when the estimate is correct

Summary

- Tradeoffs capabilities

  Sample&Collide is the most flexible algorithm considering accuracy/overhead, with its tuning parameter l that refines the estimations

  E.G for l=10: ~+/-40% accuracy & 100,000 messages

- Accuracy

  Aggregation provides a perfect estimate at the time were the process is launched,

  if a sufficient number of rounds is awaited

- Reaction to dynamicity

  Sample&Collide, Aggregation (with stabilisation observation) cope well with dynamicity.

  HopsSampling needs a little more time to converge due to averaging

Summary

- Practical considerations
  - HopsSampling is likely to be the fastest algorithm (broadcast + direct responses)
  - HopsSampling, even if lowered by the probabilistic responses, has the drawback of creating a message flood towards the initiator
  - Aggregation has the advantage of permitting the system size estimation on each node of the network, not only on the initiator

Conclusion

- New counting approaches, new sampling algorithm
- Suggestions for the use of Aggregation and HopsSampling
- **The best algorithm**? Once again, no definite answer!

Application needs dependant:

Algorithm	*Tradeoff*	*Accuracy*	*Dynamicity*	*Speed*	*Esimate availability*	*Hot Spot*
HopsSampling				+	-	-
S&C	+		+		-	
Aggregation	-	+	+		+	

Thank you!